Article

Not peer-reviewed version

Scalable *O*(*log2n*) Dynamics Control for Soft Exoskeletons

Julian D. Colorado * , Diego Mendez , Andres Gomez-Bautista , John E. Bermeo , Catalina Alvarado-Rojas , Fredy Cuellar

Posted Date: 29 September 2024

doi: 10.20944/preprints202409.2214.v1

Keywords: soft-exoskeletons; parallel computing; embedded systems; HIL; dynamics contro

Article

Scalable $O(log_2 n)$ Dynamics Control for Soft Exoskeletons

Julian D. Colorado *, Diego Mendez, Andres Gomez-Bautista, John E. Bermeo, Catalina Alvarado-Rojas and Fredy Cuellar

School of Engineering, Pontificia Universidad Javeriana, Bogota, 110231, Colombia
* Correspondence: coloradoj@javeriana.edu.co

Abstract: Robotic exoskeletons are being actively applied to support the activities of daily living (ADL) for patients with hand motion impairments. In terms of actuation, soft materials and sensors have opened new alternatives upon conventional rigid body structures. In this arena, biomimetic soft systems are playing an important role for modeling and controlling the human hand kinematics without the restrictions of rigid mechanical joints, while having an entire deformable body with limitless points of actuation. In this paper, we address the computational limitations for modeling large-scale articulated systems for soft robotic exoskeletons, by integrating a parallel architecture to compute the exoskeleton's dynamics equations of motion (EoM), obtaining a computation with $O(log_2 n)$ complexity for the highly-articulated n degrees of freedom.

Keywords: soft-exoskeletons; parallel computing; embedded systems; HIL; dynamics control

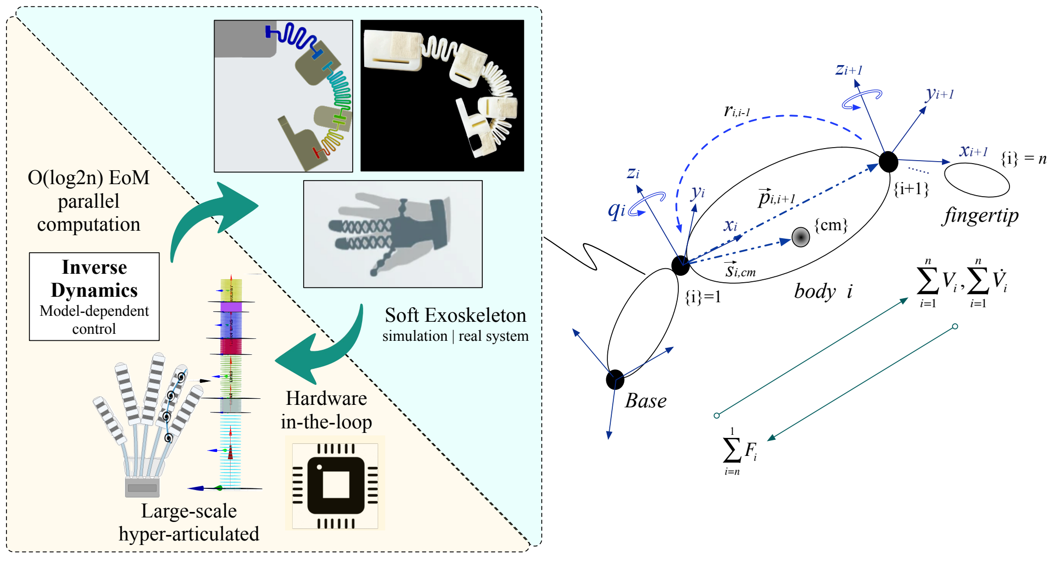
1. Introduction

Exoskeletons are emerging as effective tools for facilitating motor rehabilitation in patients with impaired upper limb function [1,2]. In this regard, robotic exoskeletons designed specifically for the hand have the potential to enhance hand rehabilitation and assist individuals in activities of daily live (ADLs). The human hand poses a significant challenge for the design and control of such devices, due to its high bio-mechanical complexity [3,4]. The hand can perform precise manipulation tasks, thanks to the coordinated action of 27 bones, 27 joints, and 34 intrinsic and extrinsic muscles.

Hand exoskeletons commonly mimic the skeletal structure of the fingers, the phalanges, by means of joint-based rigid-bodies. Recently, the use of soft materials has been proposed, as they allow for more natural movements and better adaptation to the hand dexterity. The main efforts on soft exoskeletons have focused on deformable materials and real-time control of an increased number of degrees of freedom (DoF) [5–8]. However, robust closed-loop control strategies are difficult to apply since most of these methods require the calculation of the exoskeleton dynamics to compute the control signal in real-time. Among the control models for exoskeletons, position-based and force-based controls are widely used. Additionally, control strategies based on the evaluation of electromyographic (EMG) signals, combined with machine learning algorithms are increasingly proposed [9–11]. More recently, electroencephalographic (EEG) signals and cameras have also been proposed for motor control [12]. For instance, in [13], a rigid-soft combined mechanism is developed to support hand rehabilitation based on a EEG brain-controlled switch implemented in Matlab™, while an external VICON™motion capturing system is required to determine the fingers position in 3D space. The authors in [14] present a force myography (FMG) to allow grasping assistance. The controller is based on a switching algorithm to properly modulate a soft-extra-muscle (SEM) glove, while machine-learning models running offline were trained to estimate grasp forces. Approaches in [11,15,16] depend on machine-learning offline processing to support EEG/EMG control, force-based control or simple switch-driven modulation.

In terms of actuation, pneumatics actuators (PAs) [17], electroactive polymers (EAP) [18] and shape memory alloys (SMAs) [19] are the most predominant technologies used in soft robotic exoskeleton and gloves. As mentioned in [11,18], in most of the cases, finite elements methods (FEMs) are used to model flexible and deformable properties, requiring compute-intensive requirements. In previous efforts reported in [20,21], we presented the development of a rigid-body robotic exoskeleton to assist in the recovery of hand motor function for post-stroke subjects. Our study proposed a closed-loop model predictive control (MPC) method with the aim of compensating the effects of muscle fatigue during the rehabilitation therapies. To this purpose, EMG-based muscular effort must be detected within the control loop in order to modulate the exoskeleton's velocity accordingly. Since the MPC method requires the computation of the exoskeleton dynamics model, applying this control algorithm implies significant challenges to soft exoskeleton mechanisms. In fact, few works in the literature [22] have applied robust control methods to modulate a soft robotic exoskeleton.

In this paper, we tackle the challenge of calculating the exoskeleton's equations of motion (EoM) to support robust dynamics control for a hybrid soft-exoskeleton design based on compliant joints, as depicted in the inset of Figure 1a. We have adopted the parallel structure introduced by [23] to solve large-scale EoM dynamics as a foundation of the proposed soft-based highly-redundant joint exoskeleton robot. The proposed dynamics control method via inverse-dynamics EoM is based on the well-known Recursive Newton-Euler Algorithm (RNEA), presented by [24]. Figure 1b details the multi-body diagram to represent each digit of the exoskeleton. Spatial operators for velocity (V_i), acceleration ($\dot{V}_i$) and force (F_i) led to six-dimensional physical quantities that combined the angular and linear motions of the system. The RNEA method computes two recurrences to propagate the aforementioned spatial operators from the exoskeleton's base to each fingertip of the serial-chain. In this regard, solving the inverse-dynamics via the RNEA method allows for the calculation of applied forces to properly control the exoskeleton's dynamics motion. Other works in [25–27] have used similar approaches to dynamics modeling and robust control for exoskeletons based on the RNEA method. Nonetheless, to the best of our knowledge, this is the first work to apply a parallel solution for the RNA as an alternative to control soft exoskeletons in real-time.



(a) (b)

Figure 1. (a) Hardware in-the-loop approach to compute large-scale exoskeleton's multi-body dynamics using parallel computing. (b) Soft articulated multi-body diagram.

As mentioned, classical techniques used in soft-robotics that are mostly based on finite element theory are not suitable in our application, since we require online computation of the dynamics model within a closed-loop control algorithm capable to scale up. To this purpose, hardware-based parallel computation of the EoM will be processed within an embedded system with multi-core CPU/GPU capabilities, to support the real-time calculation of the exoskeleton's large-scale dynamics model, considering up to $n = 512$ degrees of freedom to emulate continuous flexible soft bending, and achieving a $O(log_2 n)$ computational complexity. To support the evaluation of this approach, we propose a hardware-in-the-loop (HIL) architecture as represented in Figure 1a. This control scheme consists of two main components: (i) a software-side modeling the soft-exoskeleton under analysis, which is supported by Matlab®, directly running on a desktop computer; and (ii) a hardware-side supported by an embedded platform with highly parallel computing capabilities, in charge of calculating the inverse dynamics control to drive the exoskeleton.

By adopting this HIL-based architecture, it is possible to represent very complex models, such as the referenced soft-exoskeleton, without the necessity of actually building it and integrating all the required sensors and acquisition stages. Nonetheless, this approach also allows us to validate the proposed control scheme in real hardware platforms, such as the selected embedded systems. The software-side reproduces all the complex dynamics of the exoskeleton and generates the corresponding outputs (position, velocity and acceleration) that are sent to the hardware-side. With these signals coming from the exoskeleton's *digital twin*, the hardware-side applies the corresponding control scheme and in turn generates the control signals to the *virtual* actuators (e.g., motors), also implemented in the software-side. As it has been explained before, the biggest advantage is the ability to test, validate and evaluate the performance and scalability (with respect to the number of available cores or processing units) of the control scheme as it would execute in a real deployment. Our main contributions can be summarized as follows:

- an alternative approach to calculate dynamics equations of motion for soft exoskeletons in real-time, adopting a parallel computation for highly-articulated systems with a $O(log_2 n)$ computational complexity, and
- the integration and deployment of this approach on embedded systems, facilitating modular dynamics-based control of soft exoskeletons by following a hardware-in-the-loop (HIL) approach.

The rest of the paper is organized as follows: Section 2.1 briefly introduces several design approaches for the proposed soft exoskeleton, based on previous works reported in [20,21,28]. Also, the formulation for the equations of motion (EoM) based on the Newton-Euler formalism using spatial operators; Section 2.2 details the proposed parallel computing approach for the EoM, by applying the well-known RNEA method (Recursive Newton-Euler Algorithm) [24,25], allowing dynamics control of the soft exoskeleton with a computational performance $O(log_2 n)$; Section 3 presents the computing results obtained from several multi-core CPU/GPU embedded systems running under a HIL approach; Section 4 deals with discussions and remarks about the proposed approach for controlling soft exoskeletons; and Section 5 concludes this work with some insights, limitations and future work.

2. Methods

Our first design approach for the exoskeleton was presented in previous works reported in [20,21]. The system was conceived as a branched physical model, in which each finger is composed by a chain of rigid bodies serially connected through joint links. Figure 2a details the former mechanical design. As mentioned, we are working on a novel design approach supported by a hybrid soft mechanism based on compliant joints, as shown in Figure 2c. In order to apply our proposed method to model such mechanism, we require to determine a sufficient amount of joints to emulate soft motion properties. Figure 2b details this approximation to a soft finger of the exoskeleton via 512 degrees of freedom.

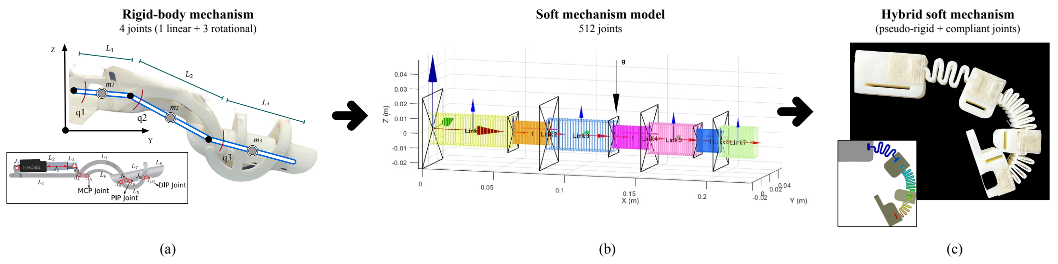


Figure 2. (a) Former rigid-body mechanism reported in previous work in [20], composed by 3 under-actuated joints connected to a single actuation input driven by a linear actuator. (b) Soft model implemented in SoRoSim©Matlab™composed by 512 degrees of freedom that emulate flexible bending [29]. (c) Proposed novel mechanism based on a semi-soft structure driven by compliant joints.

2.1. Equations of Motion

Soft motion is described based on Newton-Euler dynamics extended to large-scale articulated systems. Equations of Motion (EoM) are presented using spatial operators in $\mathbb{R}^6$, allowing joint kinematics and dynamics calculations to mimic soft deformation. Next subsection reviews the fundamental concepts of the classic mechanics that are related to body dynamics modeling, establishing an appropriate mathematical representation of the physical quantities involved in the dynamics behavior of the robotic exoskeleton.

Assuming from Figure 1b that the joint frame i and the center of mass (cm) are two points located on the body i, the term $\vec{s}_{i,cm} \in \mathbb{R}^3$ is the position vector connecting frames i and cm, while $\vec{p}_{i,i+1} \in \mathbb{R}^3$ connects consequently joint frames that are serially coupled. Also, the term $r_{i,i-1} \in \mathbb{R}^{3\times3}$ is the rotation matrix that relates frames $\{i\}$ with $\{i-1\}$. The translational (v_i) and angular velocities (ω_i) for each body, as well as and forces (f_i) and moments (τ_i) acting on the joint frame can be integrated into spatial operators that led to six-dimensional physical quantities, by combining the angular and linear variables of body motions. Using spatial notation, velocities V, accelerations $\dot{V}$, and forces F of a body i are expressed with respect to center of mas $\{i, cm\}$ using six-dimensional vectors in $\mathbb{R}^6$, as follows:

$$V_{i,cm} \equiv \begin{bmatrix} w_{i,cm} \\ v_{i,cm} \end{bmatrix}, \dot{V}_{i,cm} \equiv \begin{bmatrix} \dot{w}_{i,cm} \\ \dot{v}_{i,cm} \end{bmatrix}, F_{i,cm} \equiv \begin{bmatrix} \tau_{i,cm} \\ f_{i,cm} \end{bmatrix} \tag{1}$$

Using the position vector $\vec{s}_{i,cm} \in \mathbb{R}^3$, shown in Figure 1b, the physical quantities can be also expressed with respect to the joint frame $\{i\}$, as:

$$\begin{aligned} V_{i,cm} &= S^T_{i,cm} V_i \\ F_{i,cm} &= S^T_{i,cm} F_i \end{aligned} \tag{2}$$

Being $S_{i,cm} \in \mathbb{R}^{6\times6}$ the spatial operator connecting both frames, defined by the skew-symmetric matrix $\tilde{s}_{i,cm} \equiv \vec{s}_{i,cm}$, as:

$$S_{i,cm} = \begin{bmatrix} U & \tilde{s}_{i,cm} \\ 0 & U \end{bmatrix} \tag{3}$$

Where $U \in \mathbb{R}^{3\times3}$ is the identity operator. From Equation (2), the spatial forces acting onto the joint frame (F_i) can be derived as $F_{i,cm} = \frac{d}{dt}(I_{i,cm} V_{i,cm})$, where $I_{i,cm} \in \mathbb{R}^{6\times6}$ is the spatial inertia operator of the body i with respect to its center of mass (cm), and $V_{i,cm} \in \mathbb{R}^{6\times1}$ corresponds to the spatial velocity defined in Equation (2). Differentiating as a function of time yields:

$$\begin{aligned} F_{i,cm} &= \tfrac{d}{dt}(I_{i,cm} V_{i,cm}) \\ F_{i,cm} &= I_{i,cm} \dot{V}_{i,cm} + \dot{I}_{i,cm} V_{i,cm} \end{aligned} \tag{4}$$

Using the spatial operation for translation defined in Equation (3), spatial forces with respect to the joint frame can be calculated as:

$$\begin{aligned} F_i &= S_{i,cm} F_{i,cm} \\ &= S_{i,cm} [I_{i,cm} \dot{V}_{i,cm} + \dot{I}_{i,cm} V_{i,cm}] \end{aligned} \tag{5}$$

From Equation (2), spatial accelerations can be defined as $\dot{V}_{i,cm} = S^T_{i,cm} \dot{V}_i + \dot{S}^T_{i,cm} V_i$. Replacing into Equation (5) yields:

$$\begin{aligned} F_i &= S_{i,cm} [I_{i,cm} (S^T_{i,cm} \dot{V}_i + \dot{S}^T_{i,cm} V_i) + \dot{I}_{i,cm} V_{i,cm}] \\ &= S_{i,cm} I_{i,cm} S^T_{i,cm} \dot{V}_i + S_{i,cm} I_{i,cm} \dot{S}^T_{i,cm} V_i + S_{i,cm} \dot{I}_{i,cm} V_{i,cm} \\ &= I_i \dot{V}_i + I_i \dot{S}_{i,cm} V_i + \dot{I}_i V_i \\ &= I_i \dot{V}_i + [I_i \dot{S}_{i,cm} + \dot{I}_i] V_i \end{aligned} \tag{6}$$

Since each finger of the exoskeleton is composed by a serial chain of connected joints, spatial forces can be propagated from the fingertip ($i = n$) to the base ($i = 1$), by using specific operators for rotation and translation. Therefore, a propagated term can be included into Equation (6), such as:

$$\sum_{i=n}^{1} F_i = \underbrace{(P^T_{i,i+1} R_{i,i-1}) F_{i+1}}_{propagated} + I_i \dot{V}_i + [I_i \dot{S}_{i,cm} + \dot{I}_i] V_i \tag{7}$$

Equation (7) describes a backward propagation of spatial forces, where six-dimensional operators in $\mathbb{R}^{6\times 6}$ for rotation R and translation P are defined as follows:

$$R_{i,i-1} = \begin{bmatrix} r_{i,i-1} & 0 \\ 0 & r_{i,i-1} \end{bmatrix}, \quad P_{i,i+1} = \begin{bmatrix} U & \tilde{p}_{i,i+1} \\ 0 & U \end{bmatrix} \tag{8}$$

Where $r_{i,i-1} \in \mathbb{R}^{3\times 3}$ is the classical basic rotation matrix and $\tilde{p}_{i,i+1} \in \mathbb{R}^{3\times 3}$ is the skew-symmetric matrix of the position vector $\vec{p}_{i,i+1}$. As depicted in Figure 1b, spatial forces are calculated by following a backward propagation, initiating at the fingertip frame ($i = n$) and finalizing at the base frame of the exoskeleton's finger ($i = 1$). To this purpose, the spatial terms $P^T_{i,i+1} R_{i,i-1}$ in Equation (4) allow the propagation of the spatial forces (F) through the exoskeleton, considering the external load (F_{load}) imposed by the hand or any manipulated object. Therefore, $F_{i+1} \equiv F_{load}, \quad \forall i = n$.

Applying a similar approach, both $R_{i,i-1}$ and $P_{i,i+1}$ are also used to propagate spatial velocities (V_i) and accelerations ($\dot{V}_i$) by following a forward propagation from $i = 1 ... n$, as:

$$\begin{aligned} \sum_{i=1}^{n} V_i &= \underbrace{R^T_{i,i-1} P_{i,i+1} V_{i-1}}_{propagated} + H \dot{q}_i \\ \sum_{i=1}^{n} \dot{V}_i &= \underbrace{R^T_{i,i-1} P_{i,i+1} \dot{V}_{i-1} + \dot{P}_{i,i+1} \dot{R}^T_{i,i-1} V_{i-1}}_{propagated} + H \ddot{q}_i + \dot{H} \dot{q}_i \end{aligned} \tag{9}$$

The mathematical expressions for the Equations of Motion (EoM): spatial velocities V_i, accelerations $\dot{V}_i$ and forces F_i are presented in Equation (7) and (9) respectively, and consolidated in Algorithm 1. The inputs to the algorithm are the joint trajectories (therapy motion) determined by the terms $q_i, \dot{q}_i, \ddot{q}_i$, whereas the output is the required torque τ_i for each joint i. As mentioned, the method is divided into two recurrences: (i) a forward calculation and propagation of the spatial velocities and accelerations and (ii) a backward recurrence to calculate spatial forces and the torques acting on each joint of the fingers. Table 1 presents the list of parameters used for the EoM. Nonetheless, this $O(n)$ approach is insufficient to model large-scale soft systems composed by thousands of joints that enable body deformation, since the computing time increases linearly as a function of the number of degrees of freedom n.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 September 2024 doi:10.20944/preprints202409.2214.v

Table 1. Parameters for the EoM in Algorithm 1.

Parameter	Description
i	subscript that indicates the joint frame
$V_i, \dot{V}_i$	6D spatial velocity and acceleration of body i with respect to joint frame $\{i\}$
F_i	6D spatial force of body i with respect to joint frame $\{i\}$
H	6D vector for projection onto the axis of motion
$I_{i,cm}$	spatial inertia of body i with respect to frame $\{cm\}$
$p_{i,i+1}$	3x1 position vector that joins frame $\{i\}$ to $\{i+1\}$
$\tilde{p}_{i,i+1}$	skew symmetric matrix of vector cross product of $p_{i,i+1}$
$P_{i,i+1}$	spatial translation from joint frame $\{i\}$ to $\{i+1\}$
$q_i, \dot{q}_i, \ddot{q}_i$	joint positions, velocities and accelerations of body i with respect to $\{i\}$
$r_{i,i-1}$	3x3 basic rotation matrix that projects frame $\{i\}$ onto frame $\{i-1\}$
$R_{i,i-1}$	spatial rotation from joint frame $\{i\}$ to $\{i-1\}$
$s_{i,cm}$	3x1 position vector that joins frame $\{i\}$ to $\{cm\}$
$\tilde{s}_{i,cm}$	skew symmetric matrix of vector cross product of $s_{i,cm}$
$S_{i,cm}$	spatial translation from joint frame $\{i\}$ to $\{cm\}$
U	3x3 identity operator
$v_i, \dot{v}_i$	linear velocity and acceleration of body i with respect to joint frame $\{i\}$
$\omega_i, \dot{\omega}_i$	angular velocity and acceleration of body i with respect to joint frame $\{i\}$
τ_i	torques at joint frame $\{i\}$

Algorithm 1 $O(n)$ RNE serial computation

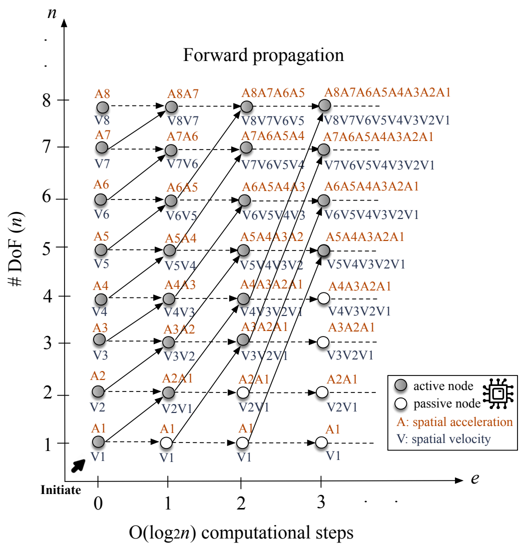
```
1. Input joint trajectory (therapy motion) for each joint $i$: $q_i, \dot{q}_i, \ddot{q}_i$
2. initialize parameters: fixed-base (no 6D motion of the wrist) and gravity acceleration acting on the carpal (along x-axis).
3. $V_0 \leftarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$,
4. $\dot{V}_0 \leftarrow \begin{bmatrix} 0 & 0 & 0 & -9.81 & 0 & 0 \end{bmatrix}^T$
5. Forward recurrence: spatial velocities and accelerations propagation from carpals to distal phalanges:
for $i = 1 \rightarrow n$ do
    (1) $V_i \leftarrow R^T_{i,i-1} P_{i,i+1} V_{i-1} + H\dot{q}_i$
    (2) $\dot{V}_i \leftarrow R^T_{i,i-1} P_{i,i+1} \dot{V}_{i-1} + \dot{P}_{i,i+1} \dot{R}^T_{i+1,i} V_{i-1} + H\ddot{q}_i + \dot{H}\dot{q}_i$
end for
----------------------------------------
6. $if$ (payload at the distal phalange is included) do $F_{i+1} \leftarrow$ payload force $F_{load}$
$else$ do $F_{i+1} \leftarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$
7. Backward recurrence: spatial forces calculated from distal phalanges to the carpals.
for $i = n \rightarrow 1$ do
    (3) $F_i \leftarrow P^T_{i,i+1} R_{i,i-1} F_{i+1} + I_i \dot{V}_i + [I_i \dot{S}_{i,cm} + \dot{I}_i] V_i$
    (4) $\tau_i \leftarrow H^T F_i$
end for
Return $\tau$
```

2.2. *$O(log_2 n)$ Parallel Computing*

Embedded systems with multi-core CPU/GPU architectures has enabled an increase in computational power that allows complex algorithms to run on the edge in real-time. This novel edge computing paradigm can be applied to our system, allowing a modular and compact design for the exoskeleton's control electronics. In this regard, we propose a reformulation of Algorithm 1 to compute the EoM in a parallel fashion.

As previously mentioned, the classical formulation of the EoM based on the spatial Newton Euler formalism requires an $O(n)$ computational complexity. It was [30] that improved the computational efficiency by computing the EoM recursively (forward dynamics) by referring the forces to generalized coordinates in the joint space. This was called the Recursive Newton-Euler Algorithm (RNEA), which was later modified by [31], by using a new approach called CRBA (Composed Rigid Body Algorithm). In [32] presented the first parallel approach to solve the CRBA for both inverse/foward dynamics problems with an $O(log_2 n)$ complexity, but strictly dependent on the overlapping costs associated to calculation of the inertial operators. In [23], the overlapping costs of Fijany's parallel approach was solved, adapting the RNEA algorithm by following the well-known Kogge and Stone recursive doubling technique [33].

Here, we adopted the solution proposed by [23], in order to solve the inverse dynamics model to compute the exoskeleton's EoM into two computational recurrences: i) a forward propagation of spatial velocities (V) and accelerations ($\dot{V}$), ii) a backward propagation of spatial forces (F). Applying the Kogge and Stone method for each recurrence yields the structure presented in Figure 3, which reduces the equation set $\forall i$ of Equation (7) and (9) for a total of $log_2 n + 1$ computational steps (e) by powers of 2. The key element for this parallel approach is to identify the $i-1$ dependent term of each equation, which can be propagated to each computer node/process of the embedded system.



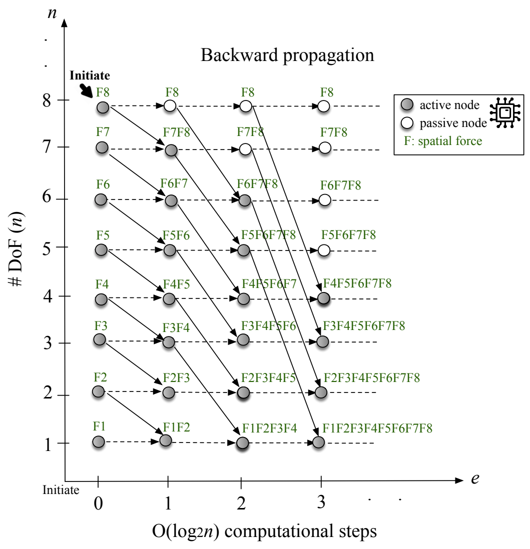


Figure 3. Parallel EoM calculation by following a $O(log_2 n)$ computational complexity. Computing steps (e) are depicted for the case of $n = 8$ degrees of freedom (DoF). For higher joints ($n >>$), the same $O(log_2 n)$ propagation scheme is maintained. In the left, both spatial velocities (V) and accelerations ($\dot{V}$) are computed by following a forward propagation. In the right, spatial forces (F) are calculated in a backward direction. Computing cores are represented by each node.

Note in Equation (9), the spatial velocity is composed by two terms: $V_i \leftarrow R^T_{i,i-1} P_{i,i+1} V_{i-1} + H\dot{q}_i$, in which V_{i-1} can be propagated whereas $H\dot{q}_i$ can be computed in parallel for each computer node at the same computational step (e). The left plot of Figure 3 illustrates this process. While the serial $O(n)$ Algorithm 1 would require $e = 8$ computational steps to solve $n = 8$ equations for the spatial velocities V_i, $\forall i = 1..n$, the parallel approach solves the same $n = 8$ equations in only $e = 3$ computational steps, since $O(log_2(8)) = 3$. Likewise, spatial accelerations follow the same structure. In the case of the spatial forces, the propagation is backwards, as detailed by the right plot of Figure 3.

Therefore, scaling up to higher degrees of freedom (n), e.g. $n = 512$ will required only $e = 9$ computational steps, assuming the embedded systems incorporates 512 processing nodes/cores (p), i.e. $n = p$. If $n > p$, multi-threading can be applied for the parallel computation of the EoM. Algorithm 2 presents the $O(log_2(n))$ parallel computation of the dynamics equations of motion for the soft exoskeleton.

In order to evaluate the performance of Algorithm 2, the Amdahl's law is used [34]. Equation (10) describes the speedup metric, which is defined by the fraction of the algorithm that is inherently serial (f_s) and the portion that is parallelized (f_p):

Algorithm 2 $O(log_2 n)$ RNE parallel computation

```
1. Input joint trajectory (therapy motion) for each joint $i$: $q_i, \dot{q}_i, \ddot{q}_i$
$\forall i$: do parallel
2. $B_{local} \leftarrow H_i \dot{q}_i, \quad M_{local} \leftarrow R^T_{i,i-1} P_{i,i+1}$
3. **Spatial velocity:** $V_i \leftarrow M_{local} V_{i-1} + B_{local}$
4. $BA_{local} \leftarrow H_i \ddot{q}_i + H_i \dot{q}_i, \quad MA_{local} \leftarrow R^T_{i,i-1} P_{i,i+1}, \quad CA_{local} \leftarrow \dot{R}^T_{i,i-1} \dot{P}_{i,i+1}$
5. **Spatial acceleration:** $\dot{V}_i \leftarrow MA_{local} \dot{V}_{i-1} + CA_{local} V_i + BA_{local}$
6. Forward recurrence: spatial velocities and accelerations propagation:
**for** $e = 1 \rightarrow log_2(n-1)$ **do**
  if $i + 2^{e-1} < n$
     if $i - 1 < 2^{e-1}$
        **send** $V_i, \dot{V}_i$ to node $i + 2^{e-1}$
     else
        **send** $M_{local}, B_{local}, MA_{local}, BA_{local}$ to node $i + 2^{e-1}$
     endif
  endif
  if $i - 1 > 2^{e-1} < n$
     if $i - 1 < 2^{e-1}$
        **receive** $V_{i-1}, \dot{V}_{i-1}$ from node $i - 2^{e-1}$
     else
        **receive** $M_{local,i-1}, B_{local,i-1}, MA_{local}, BA_{local}$ from node $i - 2^{e-1}$
     endif
  endif
  if $i > 1$
     if $e = log_2(i-1) + 1$
        $V_i = M_{local} V_{i-1} + B_{local}$
        $\dot{V}_i = MA_{local} \dot{V}_{i-1} + CA_{local} V_i + B_{local}$
     else
        $M_{local} = M_{local} M_{local,i-1}, \quad B_{local} = M_{local} B_{local,i-1} + B_{local}$
        $MA_{local} = MA_{local} MA_{local,i-1}, \quad CA_{local} = CA_{local} CA_{local,i-1}, \quad BA_{local} = BA_{local} BA_{local,i-1} + BA_{local}$

     endif
  endif
**end for**
7. $BF_{local} \leftarrow I_i \dot{V}_i + [I_i \dot{S}_{i,cm} + \dot{I}_i] V_i, \quad MF_{local} \leftarrow P_{i,i+1}{}^T R_{i,i-1}$
8. **Spatial force:** $F_i \leftarrow MF_{local} F_{i+1} + BF_{local}$
8. Backward recurrence: spatial forces propagation:
**for** $e = log_2(n) \rightarrow 1$ **do**
  if $i - 2^{e-1} > 0$
     if $i - 1 > n - 2^{e-1}$
        **send** $F_i$ to node $i - 2^{e-1}$
     else
        **send** $MF_{local}, BF_{local}$ to node $i - 2^{e-1}$
     endif
  endif
  if $i - 1 < n - 2^{e-1}$
     if $i - 1 > n - 2^{e-1}$
        **receive** $F_{i+1}$ from node $i + 2^{e-1}$
     else
        **receive** $MF_{local,i+1}, BF_{local,i+1}$ from node $i + 2^{e-1}$
     endif
  endif
  if $i < n$
     if $e = log_2(n-i) + 1$
        $F_i = MF_{local} F_{i+1} + BF_{local}$
     else
        $MF_{local} = MF_{local} MF_{local,i+1}$
        $BF_{local} = MF_{local} BF_{local,i+1} + BF_{local}$
     endif
  endif
**end for**
Return $F_i$
```

$$Speedup = \frac{1}{f_s + \frac{f_p}{n}} = \frac{1}{(1 - f_p) + \frac{f_p}{n}} \tag{10}$$

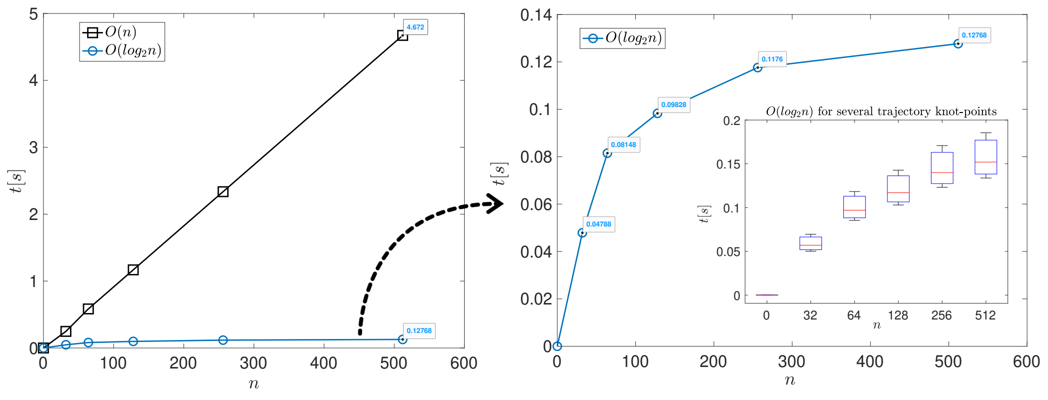
if $n = p$ the load distribution of processor (p) can be assigned to each individual joint of the exoskeleton (n: degrees of freedom). Otherwise, if $n > p$ communication packing and multi-threading processing are required.

3. Results

Algorithm 2 was coded in ANSI C programming language and implemented using the standard message-passing-interface (MPI) supported by the *mpich* library [35]. In this section, we analyze the performance of the $O(log_2 n)$ RNE algorithm running in several computing architectures:

1. Intel®Core™i7 Processor with 6 CPU cores at 2.6GHz and 512 GPU NVIDIA Quadro P1000 cores | 32GB RAM | Ubuntu 22.04.3 LTS.
2. Hardware-in-the-loop (HIL) with the Embedded Nvidia Jetson™Nano with 1024 cores at 625MHz | 8GB RAM | Ubuntu 22.04.3 LTS.

Figure 4 details the computing time response of the proposed parallel approach running on the Intel®Core™i7 with NVIDIA Quadro GPU. The plot Figure 4a compares both serial $O(n)$ and parallel $O(log_2 n)$ algorithms. As mentioned, both solutions for the inverse dynamics EoM were coded in ANSI C, using the high-performance Meschach framework; a comprehensive matrix-vector linear algebra for the operation of 6D spatial terms. The $O(log_2 n)$ RNE approach significantly outperformed the serial algorithm $\forall n > 16$ DoF, obtaining a real-time response of $t = 0.127$ in the calculation of the inverse dynamics control for the soft exoskeleton with $n = 512$ joints. In this test, we used an interpolated 3D spline trajectory with $pt = 1,000$ knot-points. Contrary, the serial $O(n)$ RNE required $t = 4.67s$ for the calculations. Considering the size of the input trajectory for the exoskeleton (pt : knot-points), the net computational time is proportional to $pt * O(log_2 n)$. In this sense, the inset within plot Figure 4b shows the resulting computing time of the parallel RNE approach with variations in the input trajectory knot-points.



(a) (b)

Figure 4. (a) Computation time comparison between the serial $O(n)$ RNE (c.f Algorithm 1) VS the parallel $O(log_2 n)$ RNE (c.f Algorithm 2). (b) closed-up to the $O(log_2 n)$ response of Algorithm 2) including computational time variations depending on the number of trajectory points (pt) defined for the exoskeleton's therapy motions. For this test, pt was adjusted between 500 up to 2,000 trajectory knot-points with a fixed step-time of 0.01s (platform: Intel®Core™i7 Processor with 512 GPU NVIDIA Quadro cores).

The overall performance of the proposed solution is presented in Figure 5. The speedup exponentially increases after $n = p = 64$, while achieving a 0.96 degree of parallelism after approximately $n = p = 256$, which demonstrates the required scalability for controlling highly-articulated and large-scale soft exoskeletons in real-time.

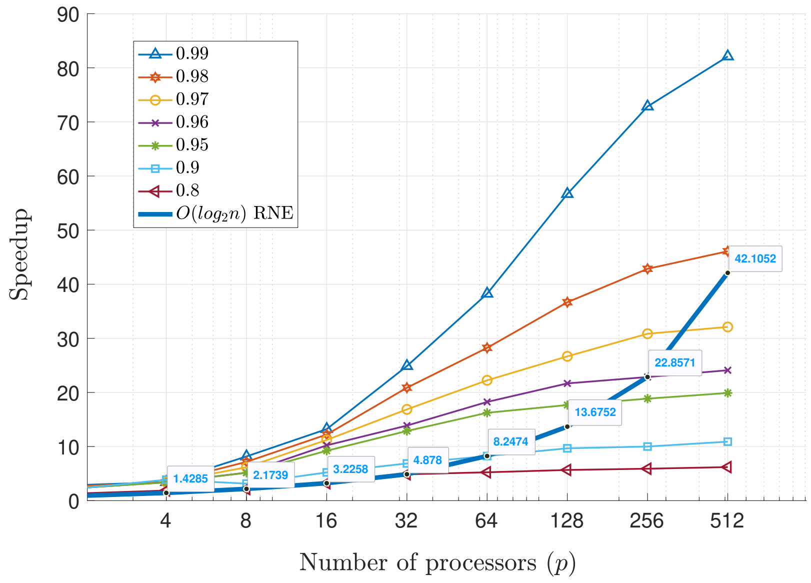


Figure 5. Speedup for the proposed $O(log_2n)$ RNE algorithm in comparison to the Amdahl's law for several degrees of parallelism from $0.8 - 0.99$ (platform: Intel®Core™i7 Processor with 512 GPU NVIDIA Quadro cores).

Another important element of the proposed $O(log_2n)$ RNE algorithm is related to the propagation structure of the EoM, in which the inherent sequential-time component of the algorithm can be drastically reduced if an embedded system-on-chip (SoC) is used. With an SoC-based embedded system, CPU/GPU cores are integrated with the memory in the same silicon, allowing higher data bandwidth to propagate data. This issue can be observed in Figure 6. Using the MPICH-library jumpshot feature, real execution with data propagation can be analyzed, by determining both serial and parallel computing time involved the calculation of the dynamics EoM. In this test, we used $n = p = 8$ to clearly visualize the computational steps (e) followed by the $O(log_2n)$ propagation of spatial velocities (V_i), accelerations ($\dot{V}_i$) and forces (F_i), as previously defined in Figure 3.

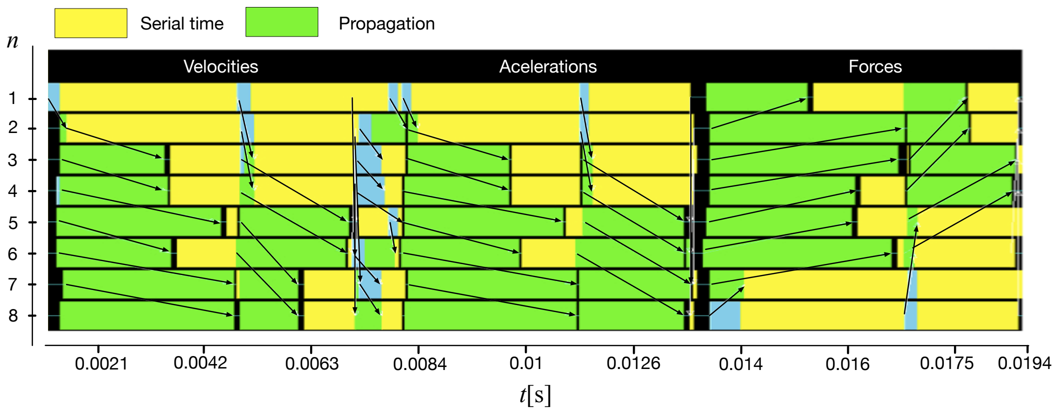
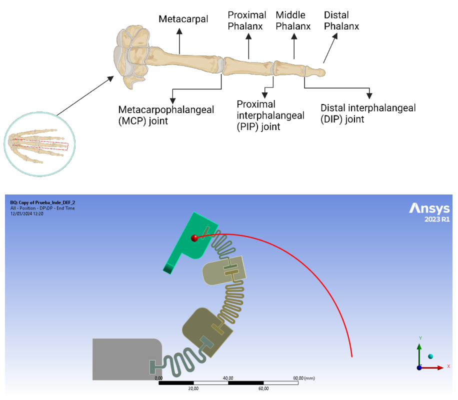


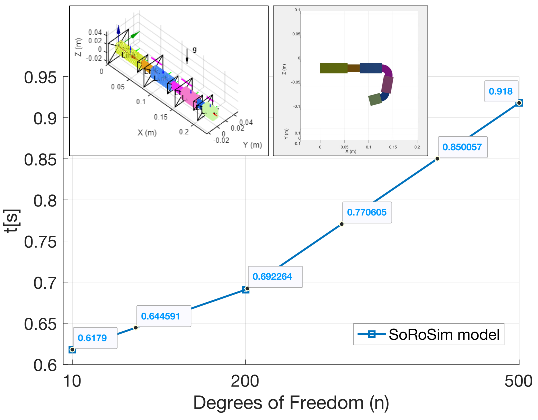
Figure 6. Computational steps of the $O(log_2n)$ RNE algorithm during execution. The propagation scheme follows the structure defined in Figure 3, detailing the MPI-driven data passing between cores ($n = p = 8$): forward propagation for spatial and accelerations, followed by a backward propagation for computing spatial forces (platform: Intel®Core™i7 Processor with 512 GPU NVIDIA Quadro cores).

As previously mentioned, classical techniques used in soft-robotics that rely on finite element theory are not suitable in our application, since we require real-time computation running online while the exoskeleton. In this regard, we compared our $O(log_2n)$ RNE model against the Ansys®simulator, performing finite element analysis based on the Newton-Raphson iteration method applied to the proposed semi-soft structure driven by the compliant joint mechanism shown in Figure 2c. Also, as detailed in the inset of Figure 7a, the exoskeleton model is composed by one finger with 3 phalanges (the distal, middle, and proximal) performing a closing/opening trajectory with 90 knot-points with a step time of 0.0075s. Under this topology Ansys used 42, 220 nodes for the semi-soft structure of the exoskeleton, requiring a simulation time of 1, 255s and execution time of 4s. Ansys determines the number of nodes and degrees of freedom (DoF) automatically. For the same trajectory with 1, 000 knot-points and a step time of 0.001s, our $O(log_2 n)$ RNE model required 0.12768s for 512 DoF, as shown in Figure 4b. Increasing up to $n = 42,220$ DoF yielded a computing time of 6.8990s, using a MPI configuration for $n > p$ (more degrees of freedom than processing cores). Even with this computing-intense configuration, our model dramatically outperformed Ansys results in terms of simulation time.

Also, Figure 7b presents computing time results for the soft model implemented in SoRoSim©Matlab™introduced by [29]. SoRoSim uses Geometric Variable Strain (GVS) approach for the dynamic simulation of soft systems. We tested several configurations from $n = 10$ up to $n = 512$ joints for the links assembly, as detailed in the insets of Figure 7b. For $n = 512$, the method performed with a computational time of 0.918s, whereas our method required 0.12768s. Besides, the computational performance of SoRoSim exhibits a linear complexity $O(n)$, i.e., higher computing times will be obtained for large-scale articulated soft systems, compromising the scalability of the solution. In contrast, our model will scale with a complexity of $O(log_2 n)$, as demonstrated in Figure 4.


Metacarpal
Proximal Phalanx
Middle Phalanx
Distal Phalanx
Metacarpophalangeal (MCP) joint
Proximal interphalangeal (PIP) joint
Distal interphalangeal (DIP) joint
Ansys 2023 R1

(a)



(b)

Figure 7. (a) Ansys®simulation for the proposed semi-soft structure driven by the compliant joint mechanism. (b) Computing time for the soft model implemented in SoRoSim©Matlab™.

Next, we present the performance of the $O(log2n)$ RNE algorithm running in a SoC-based embedded system with 1024 cores powered by the Nvidia Jetson Orin™Nano, with the ultimate goal of reducing the inherent serial time computation of Algorithm 2, and the advantage of compacting the proposed inverse-dynamics controller to support closed-loop robust control of the soft exoskeleton in real-time by following the proposed Hardware-in-the-loop (HIL) configuration. Figure 8 presents the experimental results obtained with this embedded system.

Both mpich and meschach libraries were compiled within the Nvidia Jetson Orin™Nano, allowing the computation of Algorithm 2 with similar computing times of those obtained from the Intel®Core™i7 Processor with 512 GPU NVIDIA Quadro cores, as presented in Figure 4b. For $n = p = 512$, the parallel RNE algorithm required 0.1394s for $pt = 1,000$ trajectory knot-points. As observed in Figure 8, the variance in computational time remains under 0.12s, resulting feasible for in-loop real-time response.

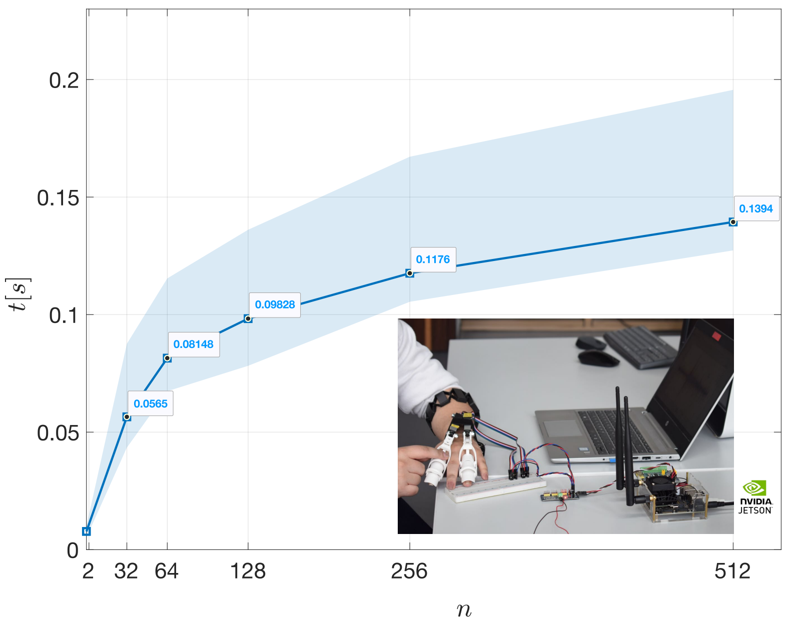


Figure 8. HIL-based $O(log_2 n)$ response of Algorithm 2 including computational time variations depending on the number of trajectory points (platform: Nvidia Jetson Orin™Nano with 1024 cores).

Finally, Table 2 presents the computing time results for the aforementioned tests: $O(n)$ refers to Algorithm 1 running on the Intel®Core™i7 Processor with 512 GPU NVIDIA Quadro cores, SoroSim refers to the tests presented in Figure 7b running under Matlab™, $O(log_2 n)$ also refers to Algorithm 2 running on the Intel®Core™i7 Processor, whereas HIL-$O(log_2 n)$ refers to Algorithm 2 running on the embedded Nvidia Jetson Orin™Nano with 1024 cores. Furthermore, for the tests conducted in Ansys®, was not possible to configure the soft exoskeleton for different DoF values, since Ansys®determined $42,220$ DoF for the assembly shown in Figure 7a, requiring a simulation time of $1,255s$ (20.91 minutes).

Table 2. Computing time comparison between platforms and methods.

Method	n=32	n=64	n=128	n=256	n=512
$O(n)$	0.24	0.58	1.16	2.33	4.67
SoroSim	0.62	0.64	0.66	0.73	0.91
$O(log_2 n)$	0.0478	0.0814	0.0982	0.1176	0.1276
HIL-$O(log_2 n)$	0.0565	0.08148	0.09828	0.1176	0.13945

4. Discussion

Algorithm 2 was implemented and tested on the embedded Nvidia Jetson Orin™Nano with 1024 cores, allowing the real-time calculation of the proposed inverse-dynamics controller based on the RNE method. As shown in Figure 8, the obtained computational complexity is of the form $pt * O(log_2 n)$, being pt the overall knot-points used to interpolate the reference trajectory for the exoskeleton.

In this regard, it is worth to highlight that those changes in the trajectory knot-points are associated with the precision and intensity of the therapy. In previous work reported in [28], we used a VICON™ motion capture system to extract the kinematics of motion associated to 6 hand gestures for rehabilitation. We implemented interpolation methods based on 3D splines to determine both cartesian and joint trajectories to drive the exoskeleton assistance. Here, the joint trajectory for positions (q_i), velocities ($\dot{q}_i$) and accelerations ($\ddot{q}_i$) are used as inputs to Algorithm 2, in order to estimate the computing time cost of the parallel inverse dynamics controller by analyzing the impact of increasing motion precision according to the computational complexity of the form $pt * O(log_2 n)$.

Although similar computing times were obtained with the embedded platform in comparison to the desktop solution (c.f Table 2), the proposed HIL-driven embedded approach allow us to deploy a modular control system, facilitating the use of the soft exoskeleton in real rehabilitation scenarios. Also, we demonstrate that even with higher values of pt, the inherent logarithmic response $log_2 n$ is maintained, supporting highly-articulated soft exoskeletons.

5. Conclusions

Classical finite element methods used in soft-robotics are not suitable for the real-time computation of model-dependent non-linear controllers, since most control algorithms require the dynamics calculation of the EoM within the closed-loop. As an alternative, our approach relies on the well-known Recursive Newton-Euler Algorithm (RNEA) calculated in a parallel fashion. As a result, a hardware-in-the-loop (HIL) structured allowed us to compute the proposed inverse-dynamics controller, by following a $O(log_2 n)$ computational performance. Real-time response was achieved by testing our model on the Nvidia Jetson Orin™Nano embedded system, enabling to close the control-loop in $139ms$ ($n = 512$) for the hybrid soft-exoskeleton mechanism depicted in Figure 2c.

The EoM presented in Algorithm 1, allows us to incorporate external forces and perturbations for the robotic exoskeleton, that can be generated by the patient during real rehabilitation scenarios. Our non-linear control method based on the inverse-dynamics RNE is able to counteract these perturbations, as shown in step 3 from Algorithm 1, being the term F_{i+1} the external payload to overcome with the control law τ_i. Here, we have demonstrated that this control output can be generated in real-time thanks to the parallel computation of Algorithm 2.

Author Contributions: Conceptualization, J. Colorado, D. Mendez; methodology, J. Colorado, D. Mendez, Andres Gomez-Bautista, John Bermeo, Fredy Cuellar; software, J. Colorado, Andres Gomez-Bautista, John Bermeo, Fredy Cuellar; validation, J. Colorado, D. Mendez, Andres Gomez-Bautista, John Bermeo, Fredy Cuellar; formal analysis, J. Colorado, D. Mendez, Andres Gomez-Bautista, John Bermeo, Fredy Cuellar, C. Alvarado-Rojas; investigation, J. Colorado, D. Mendez, C. Alvarado-Rojas; data curation, J. Colorado, Andres Gomez-Bautista; writing—original draft preparation, J. Colorado; writing—review and editing, D. Mendez, Andres Gomez-Bautista, John Bermeo, Fredy Cuellar, C. Alvarado-Rojas; supervision, J. Colorado, D. Mendez; project administration, J. Colorado; funding acquisition, J. Colorado. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the project "iREHAB: Sistema inteligente de Rehabilitación usando un Exoesqueleto para recuperar Habilidad motora en discapacidades post-ACV, usando señales Biológicas del paciente" sponsored by The Ministry of Science Technology and Innovation (MinCiencias), program 918-2022 under GRANT CTO: 622-2022, Award ID: 91805.

Data Availability Statement: Data is unavailable due to privacy or ethical restrictions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Huang, Y.; Nam, C.; Li, W.; Rong, W.; Xie, Y.; Liu, Y.; Qian, Q.; Hu, X. A comparison of the rehabilitation effectiveness of neuromuscular electrical stimulation robotic hand training and pure robotic hand training after stroke: A randomized controlled trial. *Biomedical Signal Processing and Control* **2020**, *56*, 101723.
2. Du Plessis, T.; Djouani, K.; Oosthuizen, C. A review of active hand exoskeletons for rehabilitation and assistance. *Robotics* **2021**, *10*, 40.
3. Logozzo, S.; Valigi, M.C.; Malvezzi, M. Modelling the human touch: A basic study for haptic technology. *Tribology international* **2022**, *166*, 107352.
4. Kladovasilakis, N.; Kostavelis, I.; Sideridis, P.; Koltzi, E.; Piliounis, K.; Tzetzis, D.; Tzovaras, D. A Novel Soft Robotic Exoskeleton System for Hand Rehabilitation and Assistance Purposes. *Applied Sciences* **2022**, *13*, 553.
5. Zhu, M.; Biswas, S.; Dinulescu, S.I.; Kastor, N.; Hawkes, E.W.; Visell, Y. Soft, wearable robotics and haptics: Technologies, trends, and emerging applications. *Proceedings of the IEEE* **2022**, *110*, 246–272.
6. Lin, L.; Zhang, F.; Yang, L.; Fu, Y. Design and modeling of a hybrid soft-rigid hand exoskeleton for poststroke rehabilitation. *International Journal of Mechanical Sciences* **2021**, *212*, 106831.
7. Shahid, T.; Gouwanda, D.; Nurzaman, S.G.; Gopalai, A.A. Moving toward soft robotics: A decade review of the design of hand exoskeletons. *Biomimetics* **2018**, *3*, 17.

8. Li, M.; He, B.; Liang, Z.; Zhao, C.G.; Chen, J.; Zhuo, Y.; Xu, G.; Xie, J.; Althoefer, K. An attention-controlled hand exoskeleton for the rehabilitation of finger extension and flexion using a rigid-soft combined mechanism. *Frontiers in neurorobotics* **2019**, *13*, 34.
9. Tiboni, M.; Borboni, A.; Faglia, R.; Pellegrini, N. Robotics rehabilitation of the elbow based on surface electromyography signals. *Advances in Mechanical Engineering* **2018**, *10*, 1687814018754590.
10. Sierotowicz, M.; Lotti, N.; Nell, L.; Missiroli, F.; Alicea, R.; Zhang, X.; Xiloyannis, M.; Rupp, R.; Papp, E.; Krzywinski, J.; others. EMG-driven machine learning control of a soft glove for grasping assistance and rehabilitation. *IEEE Robotics and Automation Letters* **2022**, *7*, 1566–1573.
11. Achilli, G.M.; Amici, C.; Dragusanu, M.; Gobbo, M.; Logozzo, S.; Malvezzi, M.; Tiboni, M.; Valigi, M.C. Soft, Rigid, and Hybrid Robotic Exoskeletons for Hand Rehabilitation: Roadmap with Impairment-Oriented Rationale for Devices Design and Selection. *Applied Sciences* **2023**, *13*. doi:10.3390/app132011287.
12. Tiboni, M.; Borboni, A.; Vérité, F.; Bregoli, C.; Amici, C. Sensors and actuation technologies in exoskeletons: A review. *Sensors* **2022**, *22*, 884.
13. Li, M.; He, B.; Liang, Z.; Zhao, C.G.; Chen, J.; Zhuo, Y.; Xu, G.; Xie, J.; Althoefer, K. An Attention-Controlled Hand Exoskeleton for the Rehabilitation of Finger Extension and Flexion Using a Rigid-Soft Combined Mechanism. *Frontiers in Neurorobotics* **2019**, *13*. doi:10.3389/fnbot.2019.00034.
14. Islam, M.R.U.; Bai, S. Effective Multi-Mode Grasping Assistance Control of a Soft Hand Exoskeleton Using Force Myography. *Frontiers in Robotics and AI* **2020**, *7*. doi:10.3389/frobt.2020.567491.
15. Asif, A.R.; Waris, A.; Gilani, S.O.; Jamil, M.; Ashraf, H.; Shafique, M.; Niazi, I.K. Performance evaluation of convolutional neural network for hand gesture recognition using EMG. *Sensors* **2020**, *20*, 1642.
16. Lu, Z.; Stampas, A.; Francisco, G.E.; Zhou, P. Offline and online myoelectric pattern recognition analysis and real-time control of a robotic hand after spinal cord injury. *Journal of neural engineering* **2019**, *16*, 036018.
17. Tiboni, M.; Loda, D. Monolithic PneuNets soft actuators for robotic rehabilitation: methodologies for design, production and characterization. Actuators. MDPI, 2023, Vol. 12, p. 299.
18. Peng, Z.; Huang, J. Soft Rehabilitation and Nursing-Care Robots: A Review and Future Outlook. *Applied Sciences* **2019**, *9*. doi:10.3390/app9153102.
19. Copaci, D.; Blanco, D.; Moreno, L. Wearable elbow exoskeleton actuated with shape memory alloy in antagonist movement. Proceedings of the Joint Workshop on Wearable Robotics and Assistive Devices, International Conference on Intelligent Robots and Systems, Daejeon, Korea, 2016, pp. 9–14.
20. Bonilla, D.; Bravo, M.; Bonilla, S.P.; Iragorri, A.M.; Mendez, D.; Mondragon, I.F.; Alvarado-Rojas, C.; Colorado, J.D. Progressive Rehabilitation Based on EMG Gesture Classification and an MPC-Driven Exoskeleton. *Bioengineering* **2023**, *10*. doi:10.3390/bioengineering10070770.
21. Castiblanco, J.C.; Mondragon, I.F.; Alvarado-Rojas, C.; Colorado, J.D. Assist-As-Needed Exoskeleton for Hand Joint Rehabilitation Based on Muscle Effort Detection. *Sensors* **2021**, *21*. doi:10.3390/s21134372.
22. Pal, A.; He, T.; Wei, W. Sample-efficient Model Predictive Control Design of Soft Robotics by Bayesian Optimization. *ArXiv* **2022**, *abs/2210.08780*.
23. Jaramillo-Botero, A.; Lorente, A.C.I. A Unified Formulation for Massively Parallel Rigid Multibody Dynamics of O(log2n) Computational Complexity. *Journal of Parallel and Distributed Computing* **2002**, *62*, 1001–1020. doi:https://doi.org/10.1006/jpdc.2001.1820.
24. Featherstone, R., The Recursive Newton-Euler Algorithm. In *Encyclopedia of Robotics*; Ang, M.H.; Khatib, O.; Siciliano, B., Eds.; Springer Berlin Heidelberg: Berlin, Heidelberg, 2020; pp. 1–5. doi:10.1007/978-3-642-41610-1_162-1.
25. Kvrgic, V.; Vidakovic, J. Efficient method for robot forward dynamics computation. *Mechanism and Machine Theory* **2020**, *145*, 103680. doi:https://doi.org/10.1016/j.mechmachtheory.2019.103680.
26. Chen, C.T.; Lien, W.Y.; Chen, C.T.; Twu, M.J.; Wu, Y.C. Dynamic Modeling and Motion Control of a Cable-Driven Robotic Exoskeleton With Pneumatic Artificial Muscle Actuators. *IEEE Access* **2020**, *8*, 149796–149807. doi:10.1109/ACCESS.2020.3016726.
27. Gurriet, T.; Mote, M.; Singletary, A.; Nilsson, P.; Feron, E.; Ames, A.D. A Scalable Safety Critical Control Framework for Nonlinear Systems. *IEEE Access* **2020**, *8*, 187249–187275. doi:10.1109/ACCESS.2020.3025248.
28. Arteaga, M.V.; Castiblanco, J.C.; Mondragon, I.F.; Colorado, J.D.; Alvarado-Rojas, C. EMG-driven hand model based on the classification of individual finger movements. *Biomedical Signal Processing and Control* **2020**, *58*, 101834. doi:https://doi.org/10.1016/j.bspc.2019.101834.

29. Mathew, A.T.; Hmida, I.B.; Armanini, C.; Boyer, F.; Renda, F. SoRoSim: A MATLAB Toolbox for Hybrid Rigid–Soft Robots Based on the Geometric Variable-Strain Approach. *IEEE Robotics and Automation Magazine* **2023**, *30*, 106–122. doi:10.1109/MRA.2022.3202488.
30. Amin-Javaheri, M.; Orin, D.E. Parallel Algorithms for Computation of the Manipulator Inertia Matrix. *The International Journal of Robotics Research* **1991**, *10*, 162–170, [https://doi.org/10.1177/027836499101000207]. doi:10.1177/027836499101000207.
31. Featherstone, R. A Divide-and-Conquer Articulated-Body Algorithm for Parallel O(log(n)) Calculation of Rigid-Body Dynamics. Part 1: Basic Algorithm. *The International Journal of Robotics Research* **1999**, *18*, 867–875, [https://doi.org/10.1177/02783649922066619]. doi:10.1177/02783649922066619.
32. Bhalerao, K.D.; Critchley, J.; Anderson, K. An efficient parallel dynamics algorithm for simulation of large articulated robotic systems. *Mechanism and Machine Theory* **2012**, *53*, 86–98. https://doi.org/10.1016/j.mechmachtheory.2012.03.001.
33. Kogge, P.M.; Stone, H.S. A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations. *IEEE Transactions on Computers* **1973**, *C-22*, 786–793. doi:10.1109/TC.1973.5009159.
34. Gustafson, J.L., Amdahl's Law. In *Encyclopedia of Parallel Computing*; Padua, D., Ed.; Springer US: Boston, MA, 2011; pp. 53–60. doi:10.1007/978-0-387-09766-4_77.
35. Gropp, W.; Lusk, E.; Skjellum, A. *Using MPI (2nd ed.): portable parallel programming with the message-passing interface*; MIT Press: Cambridge, MA, USA, 1999.